

Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments

Louis-Claude Canon, Emmanuel Jeannot

▶ **To cite this version:**

Louis-Claude Canon, Emmanuel Jeannot. Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments. [Research Report] RR-6476, INRIA. 2008, pp.43. inria-00261376v2

HAL Id: inria-00261376

https://inria.hal.science/inria-00261376v2

Submitted on 17 Mar 2008

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments

Louis-Claude Canon — Emmanuel Jeannot

UHP, INRIA

N° 6476

Mars 2008

Thème NUM



ISSN 0249-6399 ISRN INRIA/RR--6476--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
NANCY - GRAND EST

Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments

Louis-Claude Canon, Emmanuel Jeannot
UHP, INRIA

Thème NUM — Systèmes numériques
Équipes-Projets AlGorille

Rapport de recherche n° 6476 — Mars 2008 — 40 pages

Abstract: A schedule is said robust if it is able to absorb some degree of uncertainty in tasks duration while maintaining a stable solution. This intuitive notion of robustness has led to a lot of different metrics and almost no heuristics. In this paper, we perform an experimental study of these different metrics and show how they are correlated to each other. Additionally, we proposed different strategies for minimizing the makespan while maximizing the robustness: from an evolutionary metaheuristic (best solutions but longer computation time) to more simple heuristics making approximations (bad quality solutions but fast computation time). We compare these different approaches experimentally and show that we are able to find different approximations of the Pareto front for this bicriteria problem.

Key-words: DAG, stochastic scheduling, robustness, makespan

Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois, Campus scientifique,
615, rue du Jardin Botanique, BP 101, 54602 Villers-Lès-Nancy
Téléphone : +33 3 83 59 30 00 — Télécopie : +33 3 83 27 83 19

Évaluation et optimisation de la robustesse d'ordonnancements de graphe de tâches sur plateforme hétérogène

Résumé : Un ordonnancement est dit robuste s'il est capable d'absorber des variations dans les durées des tâches tout en maintenant une solution stable. Cette notion intuitive de la robustesse a induit beaucoup d'interprétations et de métriques différentes, mais pratiquement aucune heuristiques. Nous présentons dans cet article une étude statistique montrant comment ces métriques sont corrélées. Nous proposons ensuite différentes stratégies d'ordonnancement minimisant le temps d'éxecution et maximisant la robustesse : d'une méta-heuristique évolutionaire (bonnes solutions mais temps de calcul longs) à des heuristiques plus simple (moins bonnes solutions mais génération plus rapide). Nous comparons ces différentes approches expérimentallement et montrons que nous sommes capable de générer des approximations du front de Pareto pour ce problème bicritère.

Mots-clés : graphe de tâches, ordonnancement stochastique, robustesse, makespan

Contents

1 Introduction **4**

2 Problem description **5**
2.1 Models of Application, Platform and Execution 5
2.2 Evaluating the Makespan Distribution 6
2.3 The problem 8

3 Related Work **9**

4 Robustness Metrics **10**

5 Experimental Setup **12**

6 Empirical comparison of metrics **14**
6.1 Experimental Results 14
6.2 Discussion 18

7 A provably convergent MOEA **23**
7.1 MOEA implementation 24
7.1.1 Algorithm 24
7.1.2 Crossover and mutation operators 24
7.1.3 Evaluation 25
7.2 Convergence conditions 25
7.2.1 NSGA-II convergence 26
7.2.2 Extension to local mutation operators 27
7.2.3 Practical implications 28

8 Heuristics **28**
8.1 Aggregation principle 29
8.2 Tasks ordering 29
8.3 Assignment selection 29

9 Experimental comparison of strategies **31**
9.1 Setup 31
9.2 Search space 31
9.3 Average quality 32
9.4 Computation time 34
9.5 MOEA evolution 35

10 Conclusion **35**

11 Acknowledgement **36**

1 Introduction

The problem of scheduling an application modeled by a task graph with the objective to minimize its execution time (makespan) is a well studied problem [13, 26]. For instance, in the context of heterogeneous computing, different heuristics have been proposed in the literature to minimize the makespan such as HEFT [40], CPOP [40], hybrid remapper [30], BIL [31], hybrid method [34] or GDL [38]. However, there are a lot of other possible objectives than minimizing the makespan. Among these objectives the *robustness* has recently received a lot of attention [1, 3, 9, 14, 36, 37]. A schedule is said to be robust if it is able to absorb some degree of uncertainty in the task duration while maintaining a stable solution. The reason why robustness is becoming an important objective is the recent focus on large systems that can be dynamic and where uncertainty in terms of workload or resource usage can be very important. For instance, the duration of the tasks that compose the application and the communications between these tasks are subject to some uncertainties (due to the unpredictability of the behavior of the application and its sensitiveness of the input data). It is important to note that the robustness alone is not a metric but it gives an idea of the stability of the solution with regards to another performance metric such as schedule length, load balance of an application, queue waiting time of batch scheduler, etc. Moreover, a brief look at the literature shows that despite the fact that robustness is a very intuitive notion there is no consensus on a single metric. Conversely, almost each paper uses its own metric depending on the studied problem and the general context of the work. Furthermore, there does not exist a comparison between these different metrics, hence it is not possible to decide which metric to use when designing a heuristic.

The contribution of this paper is to provide scheduling heuristics that optimize both robustness and makespan. To achieve this goal, we have proceeded in two parts. In a first part, we focus on comparing different metrics of robustness in the context of scheduling task graph on heterogeneous systems: we model an application as a set of tasks having precedence constraints and a task as a set of statements. The performance metric we use is the makespan (the completion time of the application) and therefore, we look at the robustness of the makespan when tasks and communication may have variations (stochastically modeled by random variables) in their duration. Moreover, we try to see to which extend optimizing the makespan can help in optimizing the robustness. In other words, we try to answer the following question: *are short schedules more robust that long ones?* Therefore, the contribution of this first part is the following: we provide a comprehensive study of different robustness metrics in the case of task graph scheduling. We study how they are correlated to each other and whether robustness and makespan are conflicting objectives or not.

In a second part, based on the study of the first part and our understanding of robustness, we address our main problem: scheduling an application with the objective of minimizing the makespan and maximizing the robustness. The context of this second study is the same as the previous one: a heterogeneous environment where machines have different speeds and capabilities for which the robustness of a schedule is even harder to achieve. However designing a scheduling heuristic is a complex task. One reason is that evaluating the makespan and

the robustness in presence of stochastic variation is a #P-Complete problem[1]. Moreover, minimizing the makespan is an NP-hard problem [26]. An other reason is that in this problem the robustness and the execution time are not equivalent objectives and therefore it requires a bicriteria approach to find all the Pareto-optimal solutions (solutions that are non-dominated) while these number of solutions can be very large and NP-hard to find. Based on that remark, we propose a suit of heuristics with two goals in mind. First, help the user to choose the trade-off between the computation time and the quality of the solution. Second, help the user to choose the trade-off between robustness and makespan. For the first trade-off (quality vs. computation time) we propose heuristics going from a genetic algorithm to very fast heuristics. For the second trade-off (robustness vs. makespan) each of the proposed heuristic is multi-objective and targets the approximation of the Pareto-front (the set of Pareto-optimal solutions). Moreover, we present theoretical facts in order to prove that our multi-objective evolutionary algorithm (MOEA), directly inspired from NSGA-II [10] converges to set of Pareto-optimal solutions. Hence, it gives us a good approximation of the set of Pareto-optimal solutions.

The remaining of the paper is organized as follows. In Section 2 we present the problem and the notations used in this paper. The first part of this paper is devoted to the analysis of the robustness. Several works dealing with robustness are detailed in Section 3. The robustness metrics we use are described in Section 4. In Section 5 we present the experimental setup we used for testing and comparing the different metrics and heuristics. Comparison of the metrics are shown and discussed in Section 6. The second part of this paper is devoted to the design of bicriteria (makespan and robustness) scheduling strategies. We propose a MOEA and introduce theoretical elements for its convergence in Section 7. Our heuristics are presented in Section 8. In Section 9 we provide the experimental evaluation of the proposed heuristics. Finally, conclusion and future works are given in Section 10.

2 Problem description

2.1 Models of Application, Platform and Execution

We consider an application modeled by a stochastic task graph. A stochastic task graph is a directed acyclic graph (DAG) where nodes represent tasks and edges dependencies between these tasks. Each node and edge is valuated to represent respectively the execution cost (number of instructions) and the communication cost (number of bytes to be transmitted). To model the uncertainty *i.e.* the fact that these costs are not deterministic, we use random variables (RV) to draw these costs. Each RV gives the probability that an execution or communication cost is in a given interval.

The target platform is composed of a set of heterogeneous resources, with a complete topology, each having different capacities in terms of network and communication speed.

A schedule consists in allocating tasks to the processors respecting the precedence constraints (given by the edges of the DAG), and the resource constraints

[1] intuitively a #P problem consists in counting the number of solutions of an NP-complete problem.

(no processor can execute two tasks at the same time). In this work we consider only *eager schedule* this means that each task, once allocated to a processor starts as soon as possible in the same order that given by the schedule. This means that there is no arbitrary delay (or slack specifically added) in the schedule. Note that most of the scheduling heuristics (list, clustering, etc.) produce eager schedule.

We use the *related model* [28] to simulate the tasks execution on the resources: each CPU i is given a value τ_i, the time to execute one instruction. This means that if the cost of a task T_x drawn from its random variable is c_x the execution time of this task on processor i is $c_x\tau_i$. To compute the communication time between two tasks we proceed similarly (drawing the communication cost – number of bytes to be transmitted – from the corresponding RV and computing the transfer time using the bandwidth and latency of the link used). For instance if we assume that task x is allocated on processor i and task z is allocated on processor j then the communication cost is given by: $l_{i,j} + d_{x,y} \times S_{i,j}$, where $d_{x,y}$ is the communication volume (drawn by the random variable) between T_x and T_y, $l_{i,j}$ is the latency between processors i and j and $S_{i,j}$ is the time to send one data element between these two processors.

Since we use random variable to compute communication time and task execution time, the makespan (completion time of the schedule) of each execution of the tasks on the resources can be different: given a schedule, it is possible to have a very large number of *realizations*[2] and hence a very large number of makespans. This poses two problems.

First, a schedule usually tells when a task must start. Due to the stochastic model we use, it is not possible to ensure the start time of each task. This is why we use only eager schedules to address this problem: tasks are dynamically executed using an eager strategy where they start as soon as possible on their allocated processor while respecting the order given by the schedule.

The second problem is that we need to compute the distribution of the makespan in order to optimize our criteria. However, computing the distribution of the makespan is extremely difficult. Hagstrom [19] has shown it is a #P-complete problem. Therefore, having an accurate evaluation of this distribution is extremely costly.

We detail how to compute the makespan distribution in the next section

2.2 Evaluating the Makespan Distribution

Given a schedule S we call f_S the makespan probability density function (PDF). With f_S, one can compute the probability that the makespan M is within two bounds $[x_1, x_2]$ (noted $(\Pr[x_1 \leq M \leq x_2])$ and is given by $\int_{x_1}^{x_2} f_S(x)dx$. We will also use the cumulative distribution function (CDF) of the makespan F_S. F_S is the integral of the probability density function f_S. Therefore $F_S(x)$ gives the probability that the makespan of schedule S is lower than x (noted $\Pr[M \leq x]$).

The probability density of the makespan comes directly from the distribution of the task duration and communication time.

Computing analytically the PDF or the CDF of the makespan is not possible in the general case.

[2] a realization is computed by instantiating every computation and communication durations according to the random variables

Computing numerically the probability density or the CDF of the makespan is sometimes tractable, though computationally intensive. This is the case for task graphs with independent tasks or DAG in which the distributions are independents (an in-tree for instance). In this case (independent distributions), only two operations need to be considered (see [27, 29], for the details). The first case is when a distribution is the ancestor of another distribution. The resulting distribution is computed by adding the two distributions together. The sum of two distributions is computed by doing the convolution of the two probability density distributions and can be calculated numerically using Fast Fourier Transform (FFT) as shown in Fig. 1. The other case is when two distributions are independent and join to another one. In this case we need to compute the maximum of the two distributions. The maximum of two independent distributions is done by multiplying their CDF (see Fig. 2). Here again, it can efficiently be calculated by finding the derivative of the probability density and integrating the result.

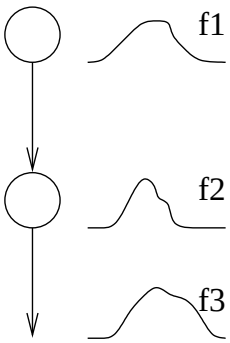


Figure 1: When two RVs follow each other the combined RV is the sum of the two RV and its PDF is computed by the convolution of the two PDFs: $f3 = f1 * f2$

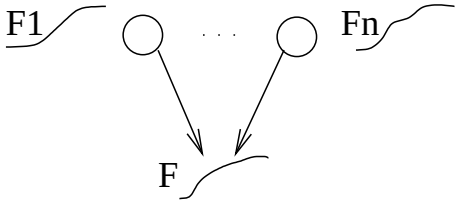


Figure 2: When n tasks merge the combined RV is the maximum of the n RV and its CDF is obtained by multiplying the n CDF together $F = F1 \times \ldots \times Fn$

In the general case, however, a DAG can have a structure such that distributions are not independent. In this case, computing the probability distribution of the makespan becomes intractable: in the general case it is #P-complete. Several authors have proposed solutions to approximate the distribution of the makespan for this case. Among these methods four are of interest for our problem.

Dodin [11] describes a method where a succession of reductions is applied to a given series-parallel graph. This results in a sole node whose random variable is equivalent to the makespan distribution of the complete graph. A mechanism is used to transform any graph into a series-parallel one with some approximation. This is one of the oldest methods and it gives acceptable accuracy.

The second method, from Spelde [29], is based on the central limit theorem which states that the sum of random variables tends to be normally distributed. Every random variable is then simplified to its unique mean and standard deviation (the only parameters needed to characterize any normal distribution) and the durations of the most critical paths in the graph are calculated. The makespan is then obtained without doing any convolution (only with maxi-

mums). Thus, this is a fast approximation method although it assumes complete independence between path durations. Refer to [29] for a description and a comparison of these methods. The above methods where designed for an unbounded number of processors. However, since the number of processors is bounded in our case, we have to modify the graph to obtain a distribution of the makespan that corresponds to a given schedule. This is done by adding edges between independent tasks when they are scheduled consecutively on the same processor (such a graph is called the *disjunctive graph*, see [37] for the details).

The third method is an improvement of the second and every random variable (from the task durations to the end time of each scheduled task) is considered as normal. The final maximum is approximated as a Gaussian distribution using Clark's moment matching approach [5]. The end time of each task is computed by firstly obtaining an approximation of the correlation between every parent of a given task, and then by applying Clark's approach. To calculate the correlation between the end times of 2 nodes, we only consider their nearest common ancestor which constitutes the main cause of the correlation. Also, to look for the nearest common ancestor, we reduce the graph to a tree by considering only the closest parent in term of Bhattacharyya distance for each task (this tree representation is only used to determine the correlation coefficient). This provides a method dealing with dependence similarly as the Dodin method (with the same order of accuracy) and having comparable computation speed as the Spelde method.

The last method is the Monte-Carlo one (see [23, 41]). This method consists in computing a great number of realizations, in order to simulate the makespan a sufficient number of times to obtain a good approximation of the distribution. Moreover, it is possible to bound the accuracy of the obtained distribution in function of the number of realizations.

2.3 The problem

Our goal is to provide scheduling heuristics that optimize both schedule length and robustness.

Once we have a distribution of the makespan we have to define which metrics have to be optimized.

Concerning the schedule length, we use the *average makespan* of the distribution. Indeed, minimizing this metrics means that on the average we minimize the overall execution time.

The problem of defining robustness has been widely studied in the literature. There exists several metrics for describing the robustness of the schedule with regards to the makespan. The next two sections are devoted to the problem of defining the robustness and comparing the different possible metrics and we will show that a good metric is the *makespan standard deviation*.

Finally, let us sum up the problem. We are given a stochastic graph and a heterogeneous environment. The goal is to schedule the tasks on the processors such that the average makespan and the standard deviation of the makespan are both minimized. Since minimizing the makespan is known to be a NP-hard problem [26] and computing the makespan distribution is #P-complete, this problem is then noted: $\mathrm{NP}^{\#\mathrm{P}}$. Moreover, in the general case, both criteria are not completely dependent and several Pareto-optimal solutions exist. Hence, it requires bicriteria scheduling strategies.

3 Related Work

We start here by covering previous work on robustness metrics. How to measure robustness is a subject that has not yet led to a wide accepted metric. Several works propose different ways to measure this metric. Ali, Maciejewski, Siegel and J.-K. Kim [1] do a good job in defining how to measure robustness: 1) defining the performance features that need to be robust, 2) identify the parameter that impacts the robustness 3) Identify how a modification of these parameters impact on the performance features 4) Identify the smallest collective variation of the parameters that make the performance features to violate acceptable variation. With this methodology the authors define a metric called the *robustness radius* that is the smallest variation of the parameters that make the performance features to exceed tolerable variation. In our case (scheduling tasks on heterogeneous system), the performance metric is the makespan, the parameters that impacts the robustness are the duration of each task and each communication. Hence, a schedule is said more robust than another one if it requires a greater change of the task duration to exceed some given bounds. The problem of that definition is that it is hard to take into account the fact that some change in task or communication duration are more likely to occur than others. Moreover, computing this metric requires a lot of effort and depends on the studied system.

In order to simplify the computation of the robustness, England, Weissman and Sadagopan [14] propose to use the Kolmogorov-Smirnov (KS) distance between the CDF of the performance metric under normal operating condition and the CDF of the same performance metric when perturbations occur. The idea is that if the KS distance is large (close to 1) this means that the two distributions are different and thus, that perturbation has a large impact on the behavior of the studied system. However, in many cases, the performance metric under normal operating condition has only one value (think for instance of the arrival time of the train at a station). In this case the distribution is a Dirac delta function and the CDF is a *step* function. Moreover, if this value is computed using the minimum of each intermediate event, the KS distance is always 1 whatever the way you organize the system. This means that this metric is not well adapted to the case where the performance metric has only one possible value, which is the case for the scheduling problem studied here.

In [36] a subset of the authors of [1] proposes a new metric called the probabilistic metric. It is defined as the probability that the performance metric is confined within a given interval. They evaluate this metric against the robustness radius (called the deterministic robustness in the paper) and show that the probabilistic metric is preferable to the deterministic metric in the case of independent tasks scheduling.

Other definitions of the robustness are available in the literature. Bölöni and Marinescu [3] propose to use the slack as a robustness metric. The slack of a task represents a time window within which the task can be delayed without affecting the makespan. The same authors suggest also to use the entropy of the performance metric distribution to compare schedules with the same makespan: given two schedules with the same makespan they conjecture that the one with the smallest entropy is the most robust. In [37] the authors study another definition of slack and show that it is equivalent to the definition given in [3]. They propose two new robustness metrics for the scheduling problem. One

is based on the average delay between the expected makespan and different realizations of the schedule under perturbation and the other is the ratio of realization that are late compared to the expected makespan. Moreover the authors show that minimizing the makespan is a contradictory objective with the problem of optimizing the robustness.

This brief look at the literature on robustness metrics shows that there is no consensus. This exemplifies the need for a comparison and a systematic study of different metrics in order to determine how these metrics are correlated to each other.

On the other side, few works had been done on robust scheduling. Most existing proactive methods are based on the insertion of slack or safety lead-time [9]. In [17], Gerasoulis *et al.* have studied the stability of DSC when task and communication durations are subject to bounded uncertainties. In Chapther 4 of [35], the sensitivity analysis of convex clustering is performed in the case of disturbance of computation durations. However, these two works use less general models than our stochastic graph one. S.-J. Kim *et al.* [24] propose a scheme for digital integrated circuit sizing problems. In their approach, every statistic duration of the problem is reduced to a deterministic value and the problem is then solved. Their way to deal with uncertainty is to obtain this deterministic value by adding the mean and a multiple of the standard deviation of the stochastic duration.

Other works consider making use of possibility theory [12] that is more adapted to model imprecision. Fargier, Fortemps and Dubois [15] apply fuzzy logic to PERT because the evaluation of the makespan possibilistic distribution is simpler than the evaluation of the probability distribution. They proposed to use pessimistic decision rules to obtain robust scheduling, however it cannot give the same insight than using usual stochastic evaluation and misses thus some stochastic aspects.

Some works have been conducted in similar scheduling areas [7, 43] but address other problems and have different models. Many proactive and reactive methods for dealing with uncertainty are surveyed in [8, 20, 21].

4 Robustness Metrics

As there is no consensus on a good metric definition, we will compare some metrics proposed in the literature to each other. However, not every metric is easy to implement. In our case we consider the makespan as the performance metric. This means that, for our problem, the robustness we measure is the stability of the makespan whatever the different realizations of the same schedule we can have. Given a task graph and a target environment, we will schedule the tasks and compute the makespan distribution. Let f be the PDF of the makespan of the schedule, F the CDF of the makespan of the same schedule and $E(M)$ the expected makespan (the average value of the makespan). Based on these definitions we define the following robustness metrics.

- **Makespan standard deviation**. Intuitively the standard deviation of the makespan distribution tells how narrow this distribution is. The narrower the distribution, the smaller the standard deviation is. This metric is related to the robustness because when you are given two schedules the one for which the standard deviation is the smaller is the one for which

realizations are more likely to have a makespan close to the average value. Mathematically we have:

$$\sigma_M = \sqrt{E(M^2) - E(M)^2}$$

- **Makespan differential entropy**. Measuring the differential entropy of a distribution to assess the uncertainty of that distribution was proposed by Bölöni and Marinesco [3]. If there is less uncertainty there is more chance than two realizations give a close result and hence that the schedule is robust.

$$h(M) = -\int_{-\infty}^{+\infty} f(x) \log f(x) dx$$

- **Slack mean**. Bölöni and Marinesco also defined the slack that gives the sum of spare time in the schedule. It is intuitively related to the robustness of the makespan as a schedule with a large slack is able to absorb a lot of uncertainty. For a deterministic schedule the slack is defined as

$$S = \sum_{i \in V} M - \mathrm{Bl}(i) - \mathrm{Tl}(i)$$

  Where M is the makespan, $\mathrm{Bl}(i)$ is the bottom level of task i (the length of the longest path from i to an exit node including i) and $\mathrm{Tl}(i)$ is the top level of node i (the length of the longest path from an entry node to node i excluding i). In our case we have random variables that define tasks and communications duration. Hence, we compute the expected value of the slack in the same way we do it for the makespan.

- **Probabilistic metric**. This metric has been defined by Shestak, Smith, Siegel and Maciejewski [36] and gives the probability that the makespan is within two bounds. If this probability is high, this means that the makespan of a given realization is likely to be close to the average makespan and hence that the robustness is high. We propose two variants of this metric. An absolute probabilistic metric that measures the probability of the makespan to be within $[E(M) - \delta, E(M) + \delta]$ where $E(M)$ is the average makespan and δ a positive constant given by the user. We also propose the relative metric that measure the probability of the makespan to be within $[E(M) \times \frac{1}{\gamma}, E(M) \times \gamma]$, where γ is a real number greater than 1. Formally, The absolute probabilistic metric is defined as:

$$A(\delta) = \Pr\left[E(M) - \delta \leq m \leq E(M) + \delta\right]$$

  and the relative probabilistic metric is defined as:

$$R(\gamma) = \Pr\left[\frac{E(M)}{\gamma} \leq m \leq \gamma \times E(M)\right]$$

- **Lateness likelihood**. A schedule is said late if its makespan exceeds a given target such as the average makespan. The lateness likelihood, or miss ratio, is defined in [37] as the probability to be late. If this metric is large this means that the makespan tends to be often late and then that the robustness is low. It is defined as:

$$L = \Pr[M > E(M)]$$

- **Makespan 0.99-quantile**. Given a schedule, this metric measures the worst makespan it is possible to have in 99% of cases. This is a hybrid criterion between the efficiency and the robustness.

5 Experimental Setup

Our study is mostly empirical and takes place in a stochastic context. Moreover, the task graphs generation, the heterogeneous platform, the stochastic simulation require each a set of parameters. Therefore, considerable attention has been given to validate the methodology, which involves the selection of relevant instances for the problem, the metrics evaluation and the exploitation of the results.

Task graphs are generated from the *Strassen* algorithm description and randomly in two ways accordingly to Tobita and Kasahara [39], namely *samepred* (each created node can be connected to any other existing nodes) and *layrpred* (nodes are arranged by layers). Every parameter used to settle tasks graphs are shown in Table 1 alongside with the selected values. Each value in bracket correspond to a single experiment while the values outside are the default ones, leading thus to 150 different experiment scenarios. For each kind of graph, we vary the number of tasks (TaskNumber), the execution and communication average costs (ExeCost and CommCost), the average number of edges per node (Avg Edge/Node), the distributions and the associated uncertainty level (UL, the ratio between the maximum and the minimum of a RV or between the 0.999-quantile and the 0.001-quantile when the previous values are inexistent). Additionally, we change the seed to obtain different graphs (SeedApp). Finally, we model heterogeneity by using the coefficient-of-variation that defines a ratio between the mean and the standard deviation of a given value in order to have a relative dispersion metric (see [2] for more details). In our case, we apply a Gamma distribution to obtain the values inside each given graph. Each parameter susceptible to change comes with a coefficient-of-variation (denoted by the prefix "V_"). Some of the parameters are ignored for Strassen graphs: the communication cost (it is already induced by the number of tasks and by the execution cost), V_Cost (the coefficient of variation associated with these 2 costs is zero) and the average number of edges per node. Besides, the number of tasks is instead: 23, 163 and 1143. The distributions of the costs in the task graphs follow either a Beta, an exponential or a normal distribution. The Beta distribution parameters we select are such that the probability distribution corresponds to our observations and expectations. To this purpose, we need a well-defined nonzero mode (implying $\alpha > 1$) and more small values than large ones (meaning we should have a right-skewed probability distribution and thus $\beta > \alpha$). Therefore, we select $\alpha = 2$ and $\beta = 5$.

The parameters for the platform generation are such that it corresponds to realistic situations. Hence, we modify the number of processors (ProcessorNumber), the bandwidth and the latency of the links (LinkBandwidth and LinkLatency) and the power of the processors (MachPower). The values are represented in Table 2.

On the scheduling side, random schedules are created by repeating iteratively the following three phases: 1) choose randomly a task among the ready ones,

Parameter	default { experiment }
GraphType	samepred layrpred strassen
TaskNumber	1000 { 10 100 1000 }
SeedApp	0 { 1 2 3 4 5 6 7 8 9 }
ExeCost (FLOP)	100 M { 10 M 1 G }
CommCost (B)	100 k { 10 k 1 M }
V_Cost	0.5 { 0.001 0.1 0.3 1 2 }
Avg Edge/Node	3. { 1. 5. }
Distribution	BETA { EXP NORMAL }
UL	1.1 { 1.0001 1.2 1.5 2 3 5 }
V_UL	0.3 { 0.001 0.1 0.5 1 2 }

Table 1: Task graph parameters

Parameter	default { experiment }
ProcessorNumber	50 { 25 100 }
SeedPlat	0 { 1 2 }
MachPower (FLOPS)	2.5 G
LinkLatency (ms)	0.1
V_Plat	0.5 { 0.001 0.1 0.3 1 2 }
LinkBandwidth (B/s)	50 M
V_LinkBandwidth	1 { 0.001 0.1 0.3 0.5 2 }

Table 2: Platform parameters

2) assign it to a randomly selected processor and schedule it eagerly, 3) update the list of ready tasks.

The evaluation of the makespan distribution (needed for most of the metrics presented in section 4) is realized with a MC (Monte Carlo) method. For each random variable (RV), we use random number generators from the GSL library [18]. We have to define the number of MC simulations required to achieve a meaningful precision. If we assume that the makespan distributions are normal, the theory of statistics says that 20,000 MC simulations are needed in order to have less than 5% of precision with a confidence level of 99% for both criteria (750,000 simulations are necessary to have a precision less than 1%, which is too much time-consuming). We obtain finally a empirical cumulative distribution function (ECDF).

Many metrics calculations are based on the makespan distribution and most are straightforward to compute from the ECDF. However, the differential entropy presents some issues. An obvious lower bound is $-\infty$ and an upper bound is proposed by Learned-Miller and DeStefano [25]. For 20,000 MC simulations, this last method is close to the true value but is highly time-consuming. Therefore, we chose an approximation method which gives close value to this bound. For the *probabilistic metric*, we have chosen $\delta = 0.1$ and $\gamma = 1.005$ in order to have values well distributed on the interval $[0; 1]$ (for different ULs, execution or communication costs than the one we used here, these values should be adapted).

In addition to robustness metrics, we also measure the following additional information, in order to study the problem structure.

- **Slack standard deviation**. The measure of the slack is similar to the measure of the makespan since the slack is also a RV. We compute thus its standard deviation.
- **Anderson-Darling statistic**. This test is one of the best ECDF omnibus tests for normality. It allows to test the normality of the makespan. The returned statistic can be though of as a distance to a normal.

On the overall we have 150 cases with different graphs type, number of nodes, target platform, uncertainty level, etc... For each generated cases, we build 5,000 random schedules. Even for the smallest graphs, the probability to get the same random schedule twice is not high and these quantities are sufficient for correlation measures. Each metric is then compared to each other visually and with the statistical Spearman correlation coefficient. This is a more robust coefficient in case of non-linear relationship between two variables than the Pearson coefficient.

6 Empirical comparison of metrics

6.1 Experimental Results

Among all the graphs we have generated, we have selected three relevant ones that are typical of the general behavior (see Figure 3, 4 and 5). On these figures, 8 metrics are compared to every other ones, leading to a matrix of 64 scattered elements. For clarifying the Figures, the *makespan 0.99-quantile* and the *lateness likelihood* are not present because they are completely equivalent respectively to the *makespan mean* and the *Anderson-Darling statistic*. On the diagonal of the matrix is given the name of each metric. On the lower part we plot the value of each metrics for the random schedules. For instance, on Fig. 3, we plot the value of the *makespan mean* against the *makespan differential entropy* on the first column and third row (the makespan is plotted on the x-axis and the entropy on the y-axis). For easing the reading of the plot, we inverted three metrics in order to have the optimization of the metrics corresponding to its minimization (hence good results should be plotted in the lower left corner of the corresponding plot). These metrics are the *slack mean*, because our initial assumption is that a robust schedule has high slack, and the two *probabilistic metrics*, since we want to maximize the probability to be in an interval. The inversion is done by multiplying by -1 the measured value that was obtained (for the slack mean) or by subtracting the measured value to 1 (for the probabilistic metrics cases). Other metrics are not inverted because optimizing them consisted already to minimize them (such as the makespan mean). Additionally, cubic smoothing spline fitting were performed on each plot, in order to visualize the correlation. The upper part of the matrix contains the value of the Spearman coefficients associated with each plot corresponding to the metrics. The higher the correlation, the closer to 1 is the absolute value of the Spearman coefficient. The minus sign for correlation means that the metrics are negatively correlated (if one metric increases, the other decreases). For instance we see on

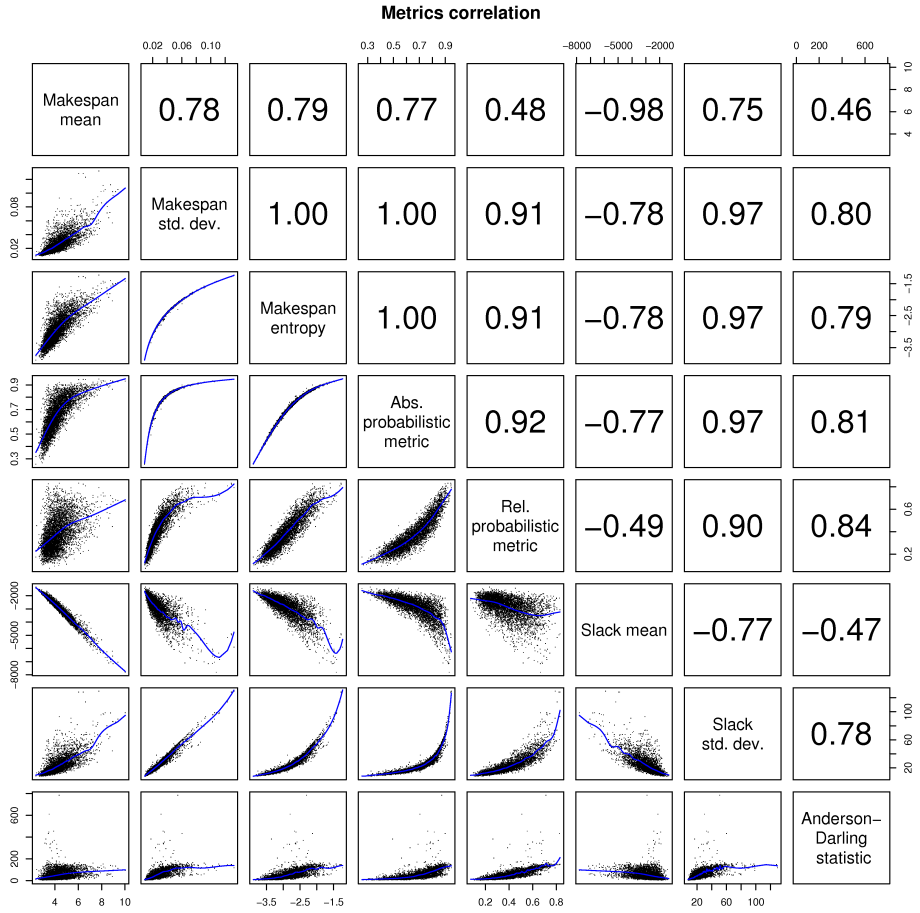


Figure 3: Metrics correlation for the *layrpred* graph with default values. Lower part of the matrix: plot for 5,000 random schedules. Upper part of the matrix: value of the Spearman coefficients.

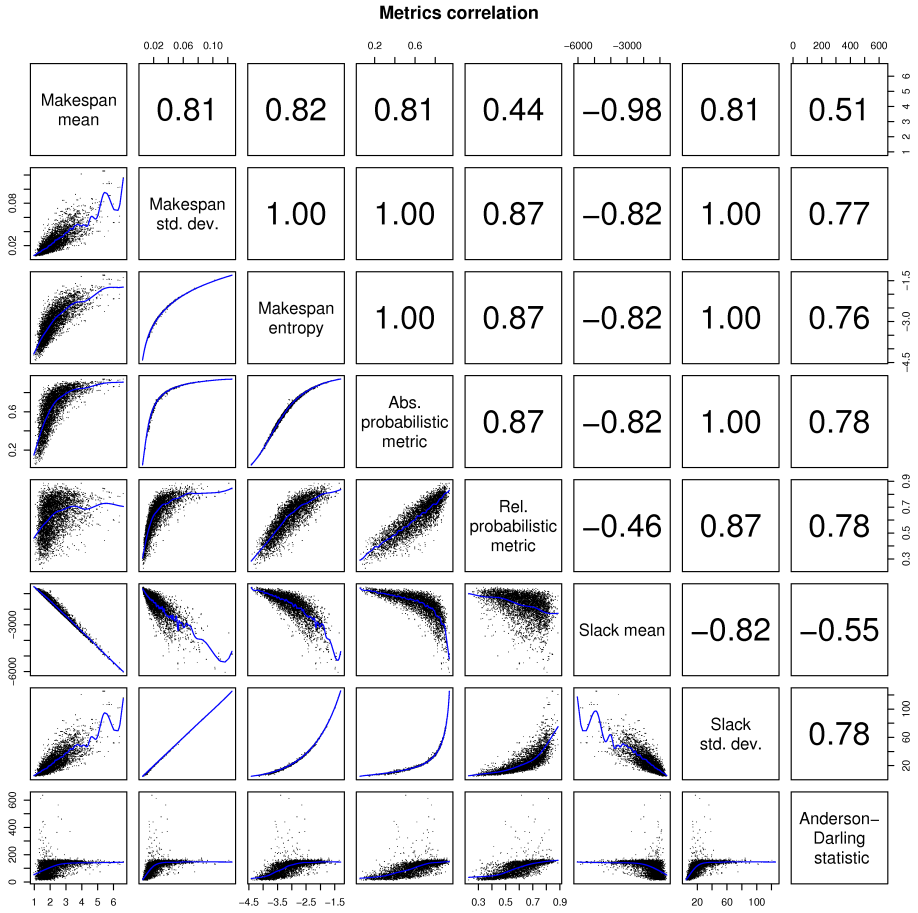


Figure 4: Metrics correlation for the *samepred* graph with default values. Lower part of the matrix: plot for 5,000 random schedules. Upper part of the matrix: value of the Spearman coefficients.

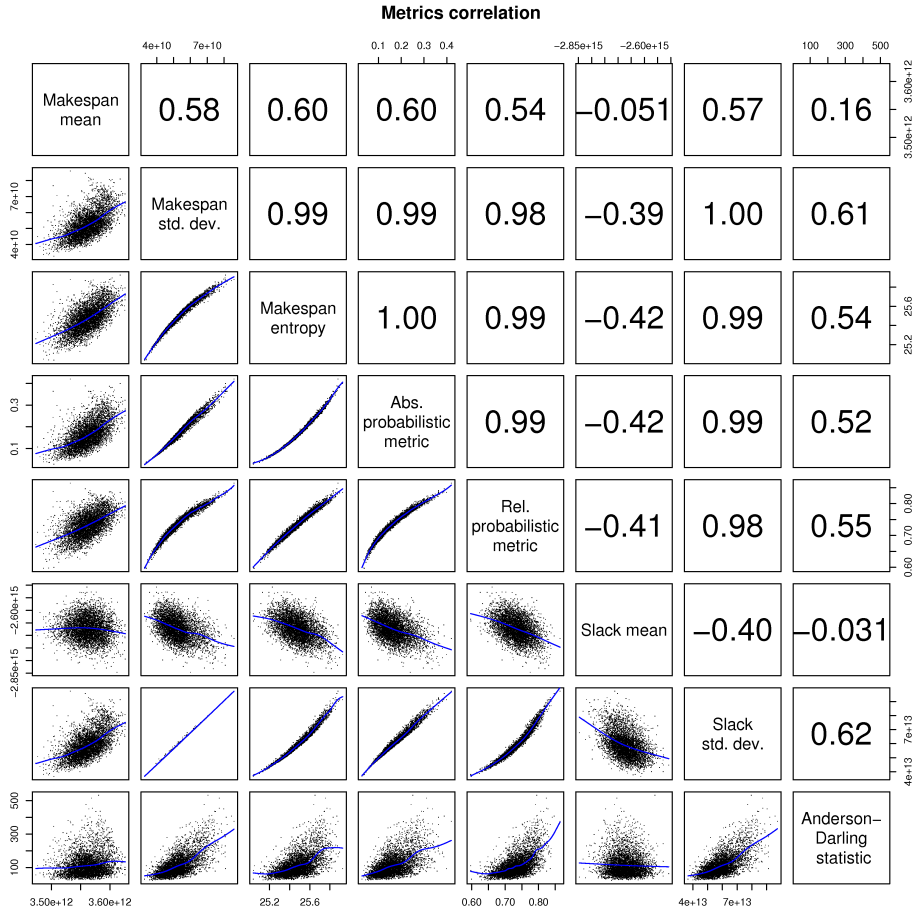


Figure 5: Metrics correlation for the *strassen* graph with a platform of 25 processors. Lower part of the matrix: plot for 5,000 random schedules. Upper part of the matrix: value of the Spearman coefficients.

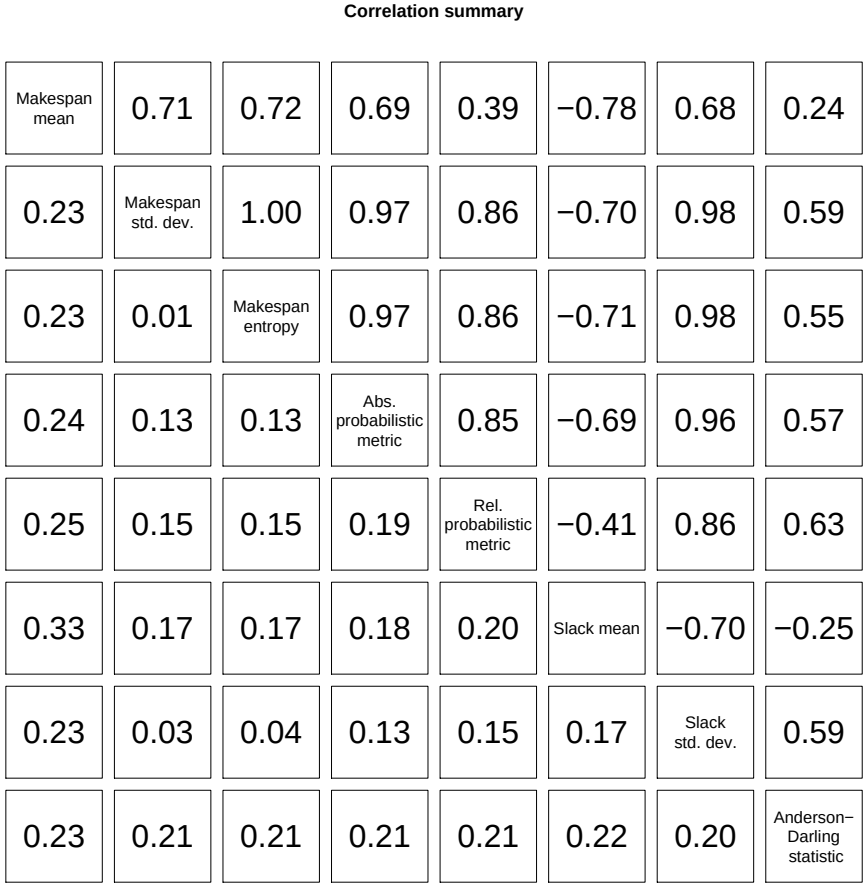


Figure 6: Average (top) and standard deviation (bottom) of the Spearman coefficients for 150 different experiments

Fig. 3 that the makespan mean and the slack mean are negatively correlated by a value of -0.98.

Since the Spearman coefficients show how the metrics are correlated to each other, they are a good way to sum-up our contribution. Hence, we have plotted in Figure 6 the matrix with the Spearman coefficients of 150 different cases. In this figure we have plotted the average value on the upper part of the matrix and the standard deviation on the lower part. We see, for instance, that the slack standard deviation and the absolute probabilistic metric are highly positively correlated (average Spearman coefficient of 0.98) with a very low standard deviation (0.04).

6.2 Discussion

We see immediately the correlation between a number of robustness metrics that are the makespan standard deviation, the differential entropy and the absolute probabilistic metric. Furthermore, the relative probabilistic metric is also cor-

related to the other ones especially in the case of *strassen* graph as can be seen in Fig. 5. As shown in Figure 6, the average Spearman coefficient is 0.86 with a standard deviation of 0.15 when compared to the makespan standard deviation. This relation is common to every generated graph, whatsoever the size, the UL (uncertainty level) or the type of graph. Indeed, the low standard deviation of the Spearman coefficient indicates that the degree of correlation is almost always the same. Then, these relationships suggest that the probability density shape remains similar for every schedule. Our explanation is based on the use of the central limit theorem which states that the sum of random variables having a finite variance (as in our case) is approximately normally distributed. Indeed, despite the fact that the makespan is obtained by performing a number of operations mixing sums and maximums, the result distribution is close to a Gaussian (however, there is a few cases where the makespan distribution fails to the normality test). This hypothesis justifies the correlation between these metrics. Since, it is a convergence result, we analyze the number of sums needed to satisfy the normal approximation in the worst case. Then, we generate a special distribution (which is constructed with a concatenation of Beta distributions, see Figure 7) and study the accuracy of the approximation. In order to evaluate this accuracy, we use the Kolmogorov-Smirnov metric (KS) that measures the maximum distance between the two CDF and a variant of the Cramér-von-Mises metric (CM) that measures the distance in terms of area. We see on Figure 8 that after only 5 sums with itself, our random variable is almost a Gaussian and that after 10, the difference is negligible. Thus, even for small graphs (with only 3 nodes on the critical path) we can simplify the robustness evaluation by calculating only one of the previously mentioned metrics, given that the PDF have convenient properties (finite expected values and standard deviations).

A second observation can be made on the relation between the makespan and the slack. On average, they are conflicting objectives in the sense that optimizing one produces a poor value for the other metric. Intuitively, a schedule having a good makespan has not much unused processor time and a schedule with a lot of slack (or spare time) is inefficient. The low correlation for *strassen* graphs is due to the existence of schedules with significant makespan and small slack (take the example where all tasks are scheduled sequentially on the same processor).

It is worth noting, too, that the makespan mean and standard deviation are correlated. Although not excellent and differing to some degree for each graph, there is a noticeable relationship in the general case. To explain this, we describe one phenomenon that arises when we are evaluating the makespan probability distribution. The variance of a random variable resulting from the sum of two others is the sum of the first two variances. If we do not considerate the implications of the maximum operator, a direct consequence is that the more tasks on the critical path, the more significant is the standard deviation, and hence, the final standard deviation is high. As we modeled the standard deviation to be proportional to the mean of task duration, heuristics producing schedules with low makespan, hence having less task or shorter tasks on the critical path, have relatively less standard deviation than schedules with large makespans. The imperfection of the correlation must be due to the maximum operations.

One surprising result is the low correlation that exists between the slack mean and other metrics, and specially the nature of this correlation. Maximizing the slack is indeed a conflicting objective with the robustness. This contradicts

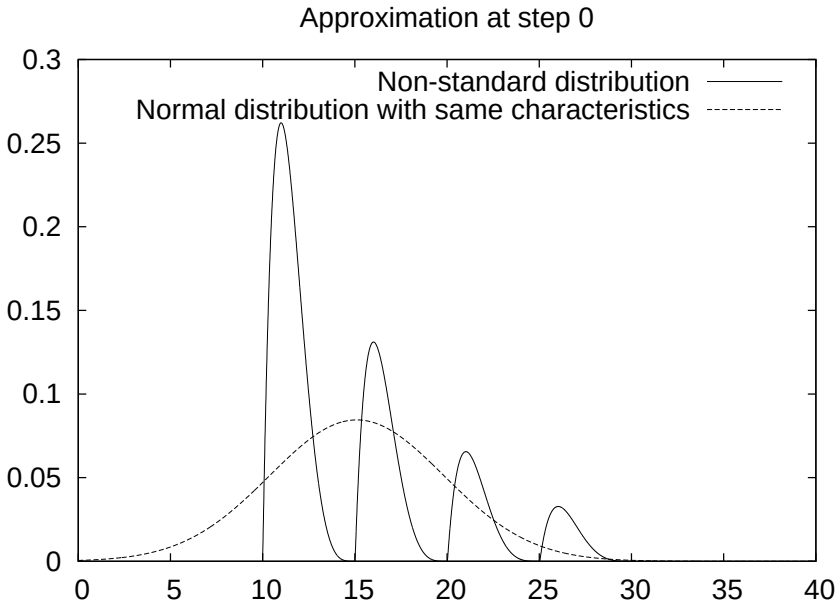


Figure 7: Two distributions (a special and a normal) having the same mean and standard deviation

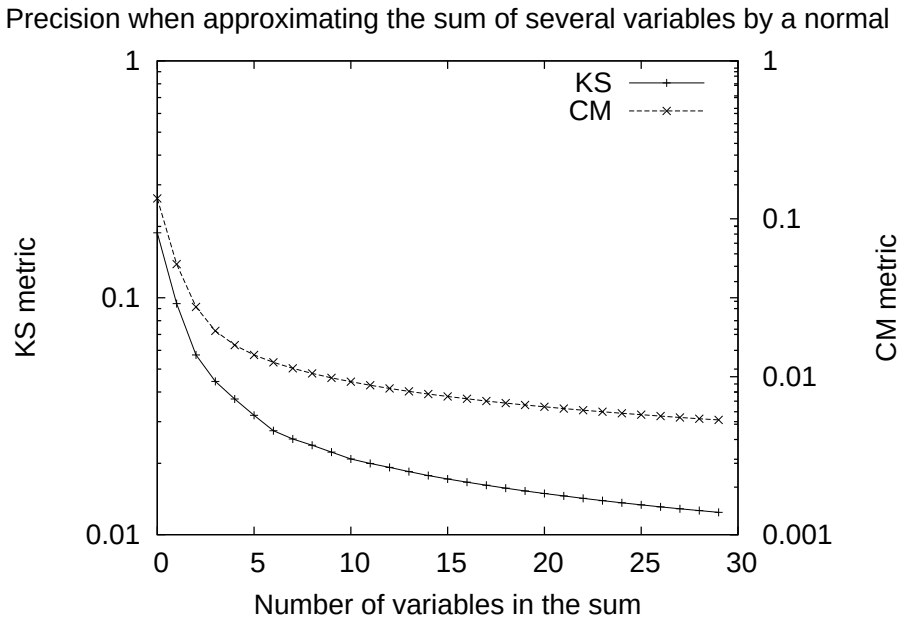


Figure 8: Precision when approximating the special distribution n-times by a normal distribution after n-sums with itself using the Kolmogorov-Smirnov and the Cramér-von-Mises metrics

the intuition that the more a schedule has slack, the more it is able to absorb uncertainty. Additionally, some previous work also proposed this metric for robustness. Hence, we present some arguments that confirm this result. Fig. 9 exhibits four examples of schedule for a join task graph of $N+1$ identical tasks having independent and identically distributed (i.i.d.) random variables. Each schedule represents different possibilities with the two objectives being the slack and the makespan standard deviation. The non-robust schedules (c and d) – according to standard deviation metric – can be interpreted as follow: almost any late task has a repercussion on the overall makespan. The schedule b) has a good robustness because only the three tasks on the critical path have an incidence on the makespan if one of those is late. The schedule a) is more subtle, because it relies on the characteristics of the maximum of two independent random variables being similar. In this case, the resulting mean is greater than the original means and more importantly, the final standard deviation is lower than at least the maximum of the two originals. A consequence is that the maximum of an infinite number of i.i.d. random variables is equal to a Dirac delta function (which is completely robust) whose value is the maximum possible value of these random variables. Then, the more tasks we are waiting for, the more we are sure that one is late, and the more the schedule is robust because we have more certainty on the expected maximum. With these four examples, we see that the slack is not necessarily related to the robustness. Moreover we see, as we already explained why, that the slack and the makespan are conflicting objectives and schedules with good makespan are often more robust. These explanations are consistent with the measures showing that slack and robustness are antagonist metrics.

The motivation behind the measure of the slack standard deviation is to provide more complete information about the slack distribution. In Figure 6, the mean and standard deviation of the slack have almost the same correlation coefficient than the mean and standard deviation of the makespan. Although less evident, the reason is the same as for the makespan. An extremely high correlation exists between the slack standard deviation and other robust metrics as the makespan standard deviation. This is because the computation of the slack consists of aggregating time intervals between task end and begin times, whose standard deviation are strongly related to the makespan standard deviation.

Since the *makespan 0.99-quantile* and the *lateness likelihood* are completely equivalent (perfect linear correlation) with the makespan mean and the Anderson-Darling statistic, respectively, they are not represented to keep the figures clear. The first equivalence is due to the fact that the variation of the makespan is not sufficient to observe a significant difference between any quantiles of the makespan distribution. The second correlation highlights a property of normal distributions, *i.e.* the nullity of the skewness which involves that the lateness likelihood of a normal is 0.5. The higher this value, the less the normality of the distribution.

It was showed in [37] that the slack was related to two robustness metrics (R1, the average lateness and R2, the lateness likelihood). However, it is conjectured that these metrics were calculated in a way that was favorable to the slack metric. Indeed, the base makespan to which was compared the makespan of realizations was obtained by simplifying each random variable to its means (which is an approximation due to the convexity of the expected value operator for the maximum of random variables). Thus R1 and R2 actually measure the

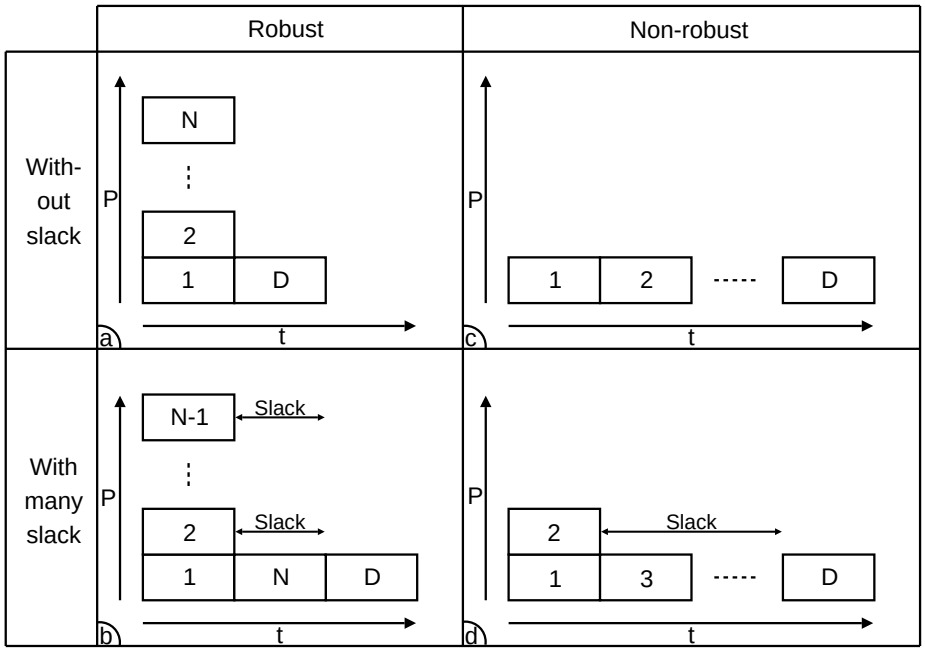


Figure 9: Four schedules on P processors for different robustness and slackness, considering i.i.d. random variables and a join graph with $N + 1$ tasks

degree of this approximation which is lowered when maximums are performed with random variables having very different means. This has more chance to happen when there is some slack. Furthermore, and this may be the main reason, the study was restricted to schedule with better makespan than the one given with the HEFT heuristic which reduces the cases of generality of the results. These hypotheses would need to be deeply examined.

Another point deserving our attention is the consequence of the maximum operator. It is stated that the maximum of two i.i.d. random variables is more robust. In our case, the random variables are not independent but the dependence depicted by the task graph does not contradict this assertion. Therefore, it implies that a way to improve robustness is to increase the similarity of the random variables on which we are making the maximum (then, equilibrating the finish time of the ancestor of every node).

Most of the above study is based on the hypothesis that the obtained makespan distributions are normal (follow a Gaussian distribution). To validate this hypothesis, we have conducted normality tests on our schedules to study at which point the normality assumption holds. The histogram of every Anderson-Darling normality statistics for the makespan distribution of random schedules is depicted in Figure 10. Half of the distributions have a statistic lower than 86 (the same as a Student t distribution with 5 degrees of freedom, which is visually quite similar to a normal, see Figure 11), and 90% of these have less than 159 and are more closer to a normal than a Weibull with parameter $\lambda = 1$ and $k = 1.84$ (Weibull are considered similar to normal for $k = 3.4$). Although it is not perfect, it is sufficient to validate the normality assumption we realize

(in the approximation scheme, for the confidence intervals and for the reduction of the robustness metrics to the standard deviation). We also notice that the choice of the distribution influences strongly the normality of the makespan distribution (it is the worst when cost distributions are exponentials).

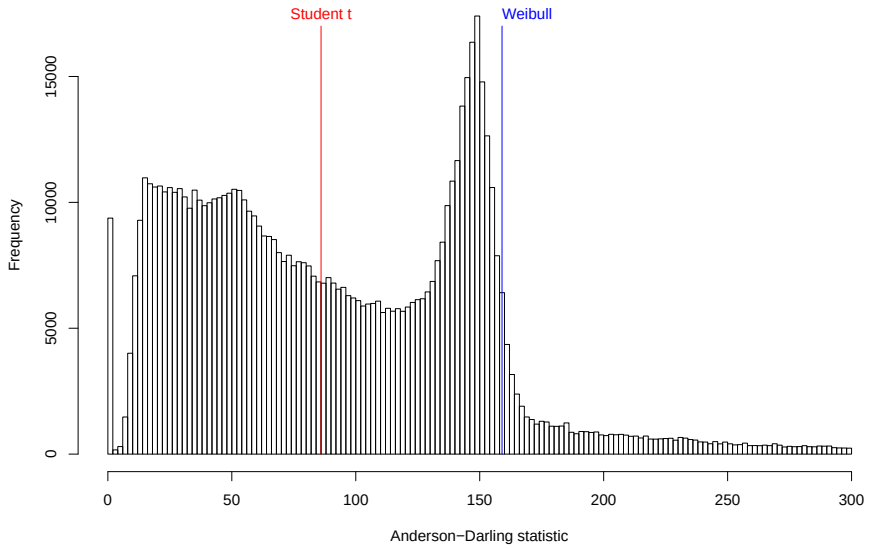


Figure 10: Histogram of Anderson-Darling statistics for the makespan distribution of random schedules

7 A provably convergent MOEA

In the previous sections we have shown that the *makespan standard deviation* is a good and easy-to-compute robustness metric. In this second part of the article, we use our understanding of robustness to design bicriteria scheduling strategies that targets optimization of the schedule length and the robustness at the same time.

In a bicriteria problem we need to find the Pareto front, which is the set of non-dominated solutions. Indeed, more than one solution can be optimal since we are dealing with a partially ordered set of solutions (two solutions are incomparable if one is better than the other for the first criterion and worse for the second).

In this section, we describe the MOEA (multi-objective evolutionary algorithm) implementation that we use and give the conditions required to guarantee its convergence. More precisely, we extend existing convergence conditions to allow the use of local mutation operators which are mutation that cannot generate any arbitrary solution of the search space in one single step.

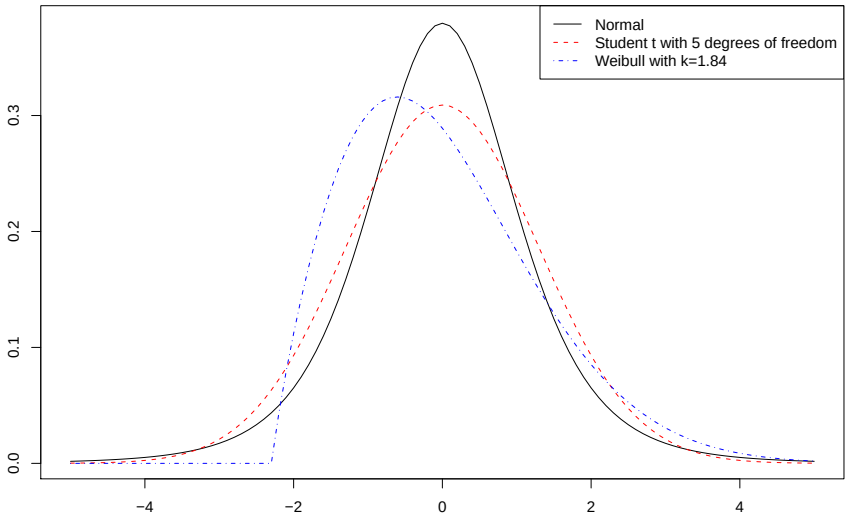


Figure 11: Three probability density functions with identical mean and standard deviation

7.1 MOEA implementation

7.1.1 Algorithm

Several successful modern MOEA exists such as NSGA-II [10], SPEA2 [45], PESA-II [6] and IBEA [44], just to mention a few. NSGA-II is the reference metaheuristic in the area and performs similarly to IBEA for combinatorial problems. Thus, we have selected the NSGA-II algorithm as it is implemented in the ParadisEO library [4].

This MOEA takes care of the multi-objective aspect and selection phase. We have still to address the crossover and mutation operators and the evaluation part.

7.1.2 Crossover and mutation operators

We use the chromosome representation and the crossover and mutation schemes described by Wang, Siegel, Roychowdhury and Maciejewski [42]. Although this mutation operator is local we show that it becomes global when applied a given number of times. This result will then be used to prove the convergence.

Since the chromosome representation consists of 2 strings, it is necessary to proceed in 2 steps. The case of the assignment string is straightforward. Each time a task is selected, a new processor is chosen randomly for it without any other constraint. Thus, the probability to get a given assignment string from an initial one with n mutation iterations is lower bounded by $\delta_{m_1} = \left(\frac{p_m}{nP}\right)^n n! > 0$, where n is the number of tasks, P the number of processor and p_m the probability that the mutation happens.

The schedule string is a linear extension of the partial ordered set (poset) E obtained from the task graph G and any of its local modifications should respect the order (corresponding to the precedence constraint). The maximum number of permutations needed to obtain any linear extension from any other is called the linear extension diameter $\mathrm{led}(E)$ and it is shown by Felsner and Reuter [16] that this diameter is upper bounded by $\mathrm{Inc}(E)$, which is the number of pairs of incomparable elements (for independent tasks, this can be up to $\frac{n(n-1)}{2}$). The probability to get a given linear extension after applying $\mathrm{led}(E)$ mutation iterations is thus lower bounded by $\delta_{m_2} = \left(\frac{p_m}{n^2}\right)^{\mathrm{led}(E)} > 0$.

Then after $M = \max(n, \mathrm{led}(E))$ mutation iterations, the resulting probability to obtain any schedule from any given one is then lower bounded by $\delta_m = \delta_{m_1}\delta_{m_2} > 0$. We can thus obtain any global mutation by successively applying the local mutation.

7.1.3 Evaluation

As stated in Section 2, the evaluation of any single schedules is #P-complete and thus an accurate evaluation is too much time-consuming. To achieve a correct precision with Monte Carlo (MC) simulations requires a lot of computation time (as shown in Section 9.5). We have thus opted for an approximation scheme: we assume that all the distributions are Gaussian and we compute the final makespan distribution by determining where independence of these distributions can be assumed. This allows fast evaluation with a correct precision in most cases.

Many sophisticated methods exist to deal with such fitness approximation in order to improve the effectiveness of the MOEA (see the survey of Jin and Branke [22]). We have, however, implemented a simple archive mechanism where the three bests non-dominated fronts are always kept. Hence, we end up with more optimized solutions with respect to the fitness approximation and this increases the probability to obtain optimal ones.

7.2 Convergence conditions

In the mono-objective case, Rudolph [32] has shown the general conditions under which an EA is guaranteed to converge (*i.e.* gives the optimal solution if applied for a sufficiently long period of time). Theorem 1 of [32] states that in order for an EA to converge to the global minimum, its Markovian kernel K should be such that, $K(x, A_\epsilon) \geq \delta > 0$ for all $x \in A_\epsilon^c = E \setminus A_\epsilon$ and $K(x, A_\epsilon) = 1$ for all $x \in A_\epsilon$, where E is the state space of the process, A_ϵ is the set of optimal states and $K(x_t, A) = \Pr[X_{t+1} \in A \mid X_t = x_t]$, namely the probability that the state of the stochastic process is in A at step $t+1$ when its state is x_t at step t. Theorem 2 of [32] gives more practical implications for the mutation and selection operators: the mutation should be *global* (any schedule can be attained from any other one in a single step with a nonzero probability) and should be followed by an elitist mechanism, *i.e.* a selection phase that preserves the best solution.

Rudolph [33] generalized these conditions to the multi-objective case. The basic assumptions of each proposition are that some degree of elitism should be present and that the stochastic process from which offspring are generated must be a homogeneous finite Markov chain with irreducible transition matrix,

roughly implying that the mutation operator should be global (the role of the crossover operator is not considered here).

We propose ourselves to adapt the convergence results for the multi-objective case to the NSGA-II algorithms and to extend the existing theory to local mutation operators having some properties we detail below.

7.2.1 NSGA-II convergence

We can obtain the convergence of NSGA-II by adapting Proposition 4 of [33]. In Algorithm 1, we describe the main loop of the algorithm which is already described in a suitable formalism in [10]. $P(t)$ and $Q(t)$ denote two sets of solutions (or two populations) at step t whose size is N. The crowding-distance allows to compare two solutions for their extent of proximity with other solutions.

Algorithm 1 NSGA-II main loop as it is described in [10] page 186

$R(t) = P(t) \cup Q(t)$	combine parent and offspring population
$\mathcal{F} =$ `fast-non-dominated-sort`$(R(t))$	$\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$ all dominated fronts of $R(t)$
$P(t+1) = \emptyset$ and $i = 1$	
until $\lvert P(t+1) + \mathcal{F}_i \rvert \leq N$	until the parent population is filled
`crowding-distance-assignment`$(\mathcal{F}_i)$	calculate crowding-distance in $\mathcal{F}_i$
$P(t+1) = P(t+1) \cup \mathcal{F}_i$	include the ith dominated front in the parent pop
$i = i + 1$	check the next front for inclusion
Sort$(\mathcal{F}_i, \preceq_n)$	sort in descending order using crowding-distance
$P(t+1) = P(t+1) \cup \mathcal{F}_i[1 : (N - \lvert P(t+1) \rvert)]$	choose the first $(N - \lvert P(t+1) \rvert)$ elements of $\mathcal{F}_i$
$Q(t+1) =$ `make-new-pop`$P(t+1)$	use selection, crossover and mutation to create a new population $Q(t+1)$
$t = t + 1$	increment the generation counter

Before formulating the appropriate facts and proposition, let us introduce some notations. f stands for the objective function, while $\mathcal{F}$ is the set of objective values. A^* is either the set of minimal elements or the set of minimal objective values of A. $B(\cdot)$ denotes the population B at any step t. Finally, the measure $\delta_B(A)$ counts the number of elements that are in set A but not in set B.

Based on Algorithm 1, we can state the following three facts:

Fact 1. *If an optimal element has entered $P(\cdot)$ it stays there forever or is replaced by another optimal element.*

Fact 2. *If $q \in Q^*(\cdot)$ and $f(q)$ dominates elements of $f(P(\cdot))$ then q moves to $P(\cdot)$ and the dominated elements leave $P^*(\cdot)$.*

Fact 3 of [33] is ignored since not used in the demonstration. Hence, the following fact corresponds to Fact 4 of [33].

Fact 3. *If there is a non-optimal element in $P^*(\cdot)$ there exists a dominating element.*

Proposition 1. *Let G be the homogeneous stochastic matrix describing the transition behavior from $P(t)$ to $Q(t)$. If matrix G is positive then $\delta_{\mathcal{F}^*}(f(P^*(t))) \to 0$ and $\lvert P^*(t) \rvert \to \min\{N, \lvert \mathcal{F}^* \rvert\}$ with probability one and in mean as $t \to \infty$.*

Proof. The proof uses the 3 above facts. See proof of Proposition 4 of [33] for the details. □

Having a global mutation followed by a binary tournament selection is sufficient to ensure the positiveness of matrix G (the matrix describing `make-new-pop`). Therefore, this implies that NSGA-II is a convergent algorithm under the assumption that mutations are global.

7.2.2 Extension to local mutation operators

We now extend the theory in the mono-objective case for assuring the convergence even with specific local mutation operators.

Let us first introduce some notations and definitions. The product kernel (K_c, K_m, K_s) represents the Markovian kernel of the entire EA process, K_c being the kernel of the crossover operator, K_m for the mutation operator and K_s for the selection. Moreover, $K^{(M)}$ is the M-th iteration of the kernel K.

We now show an auxiliary result:

Theorem 1. *Let $K_c(x, A) \geq \delta_c$ and $K_s(x, A) \geq \delta_s$ for each $x \in A$ and for each $A \subset E$. Then,*

$$(K_c\, K_m\, K_s)^{(M)}(x, A) \geq (\delta_c \delta_s)^M\, K_m^{(M)}(x, A)$$

Proof. (by induction) Let $K_c(x, A) \geq \delta_c 1_A(x)$ and $K_s(x, A) \geq \delta_s 1_A(x)$ for each $x \in E$ and for each $A \subset E$ and where $1_A(x)$ denotes the indicator function for some set A ($1_A(x) = 1$ if $x \in A$, 0 otherwise). We obtain the basis of the induction for $M = 1$ by developing $(K_c\, K_m\, K_s)(x, A)$,

$$\begin{aligned}
(K_c\, K_m\, K_s)(x, A) &= \int_E \left(\int_E K_c(x, dz) K_m(z, dy) \right) K_s(y, A) \\
&\geq \int_E \left(\int_E \delta_c 1_{dz}(x) K_m(z, dy) \right) \delta_s 1_A(y) \\
&\geq \delta_c \delta_s \int_A \left(\int_E 1_{dz}(x) K_m(z, dy) \right) \\
&\geq \delta_c \delta_s \int_A K_m(x, dy) \\
&\geq \delta_c \delta_s K_m(x, A) \qquad (1)
\end{aligned}$$

Now assume that the hypothesis is true for $M > 1$. Equation 1 induces that

$$\begin{aligned}
(K_c\, K_m\, K_s)^{(M+1)}(x, A) &= \int_E (K_c\, K_m\, K_s)^{(M)}(y, A)\, (K_c\, K_m\, K_s)(x, dy) \\
&\geq \delta_c \delta_s \int_E (K_c\, K_m\, K_s)^{(M)}(y, A) K_m(x, dy)
\end{aligned}$$

By induction hypothesis, we have

$$\int_E (K_c\, K_m\, K_s)^{(M)}(y, A) K_m(x, dy) \geq \int_E (\delta_c \delta_s)^M\, K_m^{(M)}(y, A) K_m(x, dy)$$

And by definition,

$$K_m^{(M+1)}(x, A) = \int_E K_m^{(M)}(y, A) K_m(x, dy)$$

Consequently, the hypothesis is true for $M \geq 1$. □

The main implication of Theorem 1 is that after M generations the kernel of the EA can be bounded by the M-th iteration of the mutation operator kernel. The conditions for convergence given by Theorem 1 of [32] are then fulfilled if the mutation operator becomes global when applied M times (even if it is local at each generation) and if δ_c and δ_s are strictly positive. This implies that crossover should not systematically change solutions (the probability for the crossover to be performed should then be different from 1) and that selection should not be deterministic (which is not the case for the binary tournament for example, because the worst solution is systematically removed). The EA must still have an elitism mechanism in order to keep any optimal solution. These conditions allow to leave local minimum when several local mutation steps are required for this.

7.2.3 Practical implications

In order to apply this previous extension to the multi-objective case and in particular to the NSGA-II algorithm, we need to modify it. Let demonstrate the following proposition:

Proposition 2. *Let G be the homogeneous stochastic matrix describing the transition behavior from $P(t)$ to $Q(t + M - 1)$. If matrix G is positive then $\delta_{\mathcal{F}^*}(f(P^*(t))) \to 0$ and $|P^*(t)| \to \min\{m, |\mathcal{F}^*|\}$ with probability one and in mean as $t \to \infty$.*

Proposition 2 states that the transition matrix from $P(t)$ to $Q(t + M - 1)$ should be positive. This is equivalent to say that the Markovian kernel of the EA process should be strictly bounded by 0. In section 7.1.2, we have shown that the mutation operator M-th iteration is global and thus $K_m^{(M)}(x, A) > 0$. By applying Theorem 1, we can prove the positiveness of the EA process by ensuring that the crossover does not happen systematically, and that the selection operators are not deterministic.

By inserting a stochastic decision during selection phases, we guarantee the positiveness condition. However, in this case, Fact 1 is not valid any longer. Adding an archive of fixed size containing only non-dominated elements generated so far is a sufficient mechanism to satisfy Fact 1.

To summarize, NSGA-II converges to a set of optimal elements if the mutation operator is global. Additionally, if selections are not deterministic, it converges even if the mutation is local, given that some archive mechanism exists and the crossover probability is strictly less than 1.

8 Heuristics

We introduce in this section a class of heuristics based on HEFT [40] able to generate a set of solutions intended to have good performance for both criteria from a stochastic task graph.

8.1 Aggregation principle

In order to take into account the bicriteria nature of the schedules that are constructed, we aggregate the average makespan with its standard deviation: $f(\mu, \sigma, a) = a \times \frac{\mu}{\mu_{\max}} + (1 - a) \times \frac{\sigma}{\sigma_{\max}}$, with $a \in [0, 1]$, μ and σ the current end time mean and standard deviation respectively, and $\mu_{\max}$ and $\sigma_{\max}$ the maximum mean and standard deviation of the makespan. The parameter a allows to weight each criterion accordingly to the importance we give to each one (when $a = 1$, we are only concerned by the makespan average criterion). Thus, varying a allows us to obtain different solutions.

8.2 Tasks ordering

The first part of HEFT consists of ordering the task according to their upward ranks. At this stage, we only consider average times rather than the aggregation mentioned above because it gives better results this way.

8.3 Assignment selection

We first point out that the standard deviation criterion is difficult to tackle because contrarily to the average criterion, it is non-monotonic, that is to say that the standard deviation of the end time of a given assignment can lower when tasks are added to the corresponding processor. We illustrate this in Figure 12 by scheduling a fork-join graph on 2 machines. While the end time of task 2 has a high standard deviation, the start time of task 4 is more affected by the probability density function of the end time of task 3 which has a higher mean. Hence, the overall makespan has a lower standard deviation than the partial schedule where only task 1 and task 2 are scheduled.

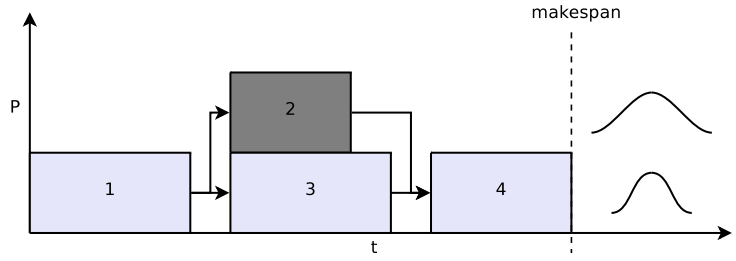


Figure 12: Schedule of a fork-join task graph (the darker the tasks, the higher their standard deviations)

We have studied several strategies and present the two most relevant. The first is based on the usual EFT policy. At each step, the schedulable task with the higher rank is selected and every possible assignment to each processor is simulated. For each assignment, the aggregate criterion defined in Section 8.1 is computed and the final assignment is the one minimizing this criterion. Figure 13 shows 2 possible assignments for a given task k where one is better for the standard deviation and the other better for the mean (the selected assignment depends on the value of a). This first heuristic scheme is called: HEFT with uncertainty level. According to how is performed the makespan distribution

evaluation we have two versions of this heuristic: *Hul_MC* when we use the Monte Carlo method and *Hul* when we use the same approximation scheme as for the MOEA.

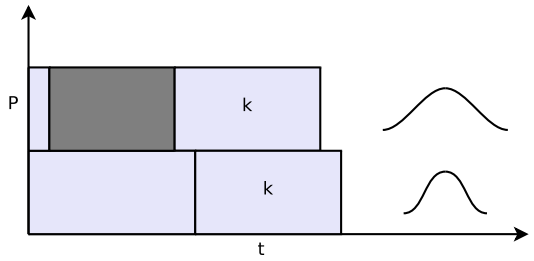


Figure 13: Two possible assignments for the schedulable task k during the scheduling process with two incomparable pairs of values (mean and standard deviation)

The second strategy is based on the observation that if a given assignment reduces (by the non-monotonicity property) significantly the standard deviation of the end time on one processor, it should be preferred to an assignment that presents the lowest standard deviation. Thus, it leads us to compute the overall maximum of every processor end time for each possible assignment and to apply the same minimization selection than before (by aggregating the mean and standard deviation of this overall maximum). Figure 14 depicts the process used for a single assignment. We introduce a dummy zero-cost task starting when all tasks are finished and compute its end time. The goal is here to apply the minimization to the assignments having significant impact on the makespan of the partial schedule rather than just to every assignment regardless of their impacts (as with *Hul*). We call it *Hulm* (HEFT with uncertainty level and maximum). Here we do not use the Monte Carlo method to evaluate the makespan distribution because we want this heuristic to be as fast as possible.

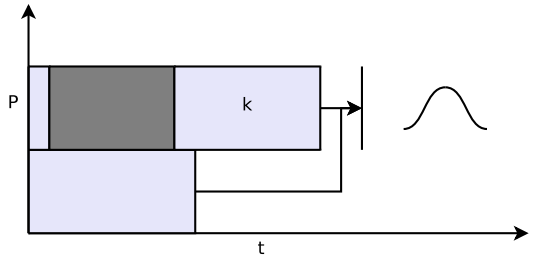


Figure 14: One possible assignment for the schedulable task k during the scheduling process with a dummy zero-cost task starting when all tasks are finished

Among other tested methodologies, we sort the aggregate criterion $f(\mu, \sigma)$ of each processor end times for each possible assignment in descending order and then select the assignment that minimizes them (using lexicographic order). More elaborated strategies along with this one generate only poor-quality

solutions and were not kept in the experiments. It testifies however the difficulty of designing relevant heuristics.

9 Experimental comparison of strategies

9.1 Setup

As mentioned in Section 6.2, we perform the robustness evaluation by calculating the standard deviation of the makespan.

We use the same experimental setup as the one describe in Section 5 for evaluating the robustness.

Concerning the MOEA setup, we consider populations of 200 chromosomes over 1,000 generations. The crossover and mutation probabilities are respectively 0.25 and 0.35. For *Hul* and *Hulm* heuristics, we vary the parameter a from 0 to 1 by step of 0.005. For *Hul_MC* which is the most costly, the step size is reduced to 0.05.

9.2 Search space

In a first attempt to study the problem specificity, we characterize the search space by generating extreme schedules present on the border fronts denoted by *SW*, *SE* and *NW* (obtained with the MOEA, when the objectives are alternatively maximized and minimized). These 3 last metaheuristics are named after the intercardinal directions (*NW* (North West) consists in minimizing the average makespan while maximizing the standard deviation). *SW* is thus designed to find optimal solutions for both criteria. Figure 15 depicts the search space of one experiment scenario. Additionally, the previous random schedules are also represented.

An immediate observation is the apparent correlation between both criteria (even the *SE* and the *NW* sets follow a linear pattern) which confirm our above remarks. Table 3 summarize the correlation coefficients over every experiment scenario (a value close to 1 implies a high linear relationship between the criteria) for the *rand* schedules. Tukey's five number summary corresponds to minimum, the first quartile, the median, the last quartile and the maximum of the set of measures. We see that 25% of the correlation coefficients of Strassen graphs are lower than 0.78 and 75% are higher. We observe that schedules for Strassen graphs have highly correlated means and standard deviations.

Graph	Min	25%	Med	75%	Max
STRASSEN	0.26	0.78	0.81	0.81	0.91
LAYRPRED	0.20	0.49	0.57	0.73	0.80
SAMEPRED	0.095	0.31	0.40	0.76	0.81

Table 3: Tukey's five number summary of correlation coefficients by graph kind

It is also worth noting that the *SW* front is almost always isolated from other regions and has a limited spread. Pareto-optimal solutions are hence quite close in regards to the global search space. When V_UL > 0.3, correlations are however the worst and the *SW* fronts are larger. Therefore, minimizing the average makespan does not necessarily induce good robustness when V_UL

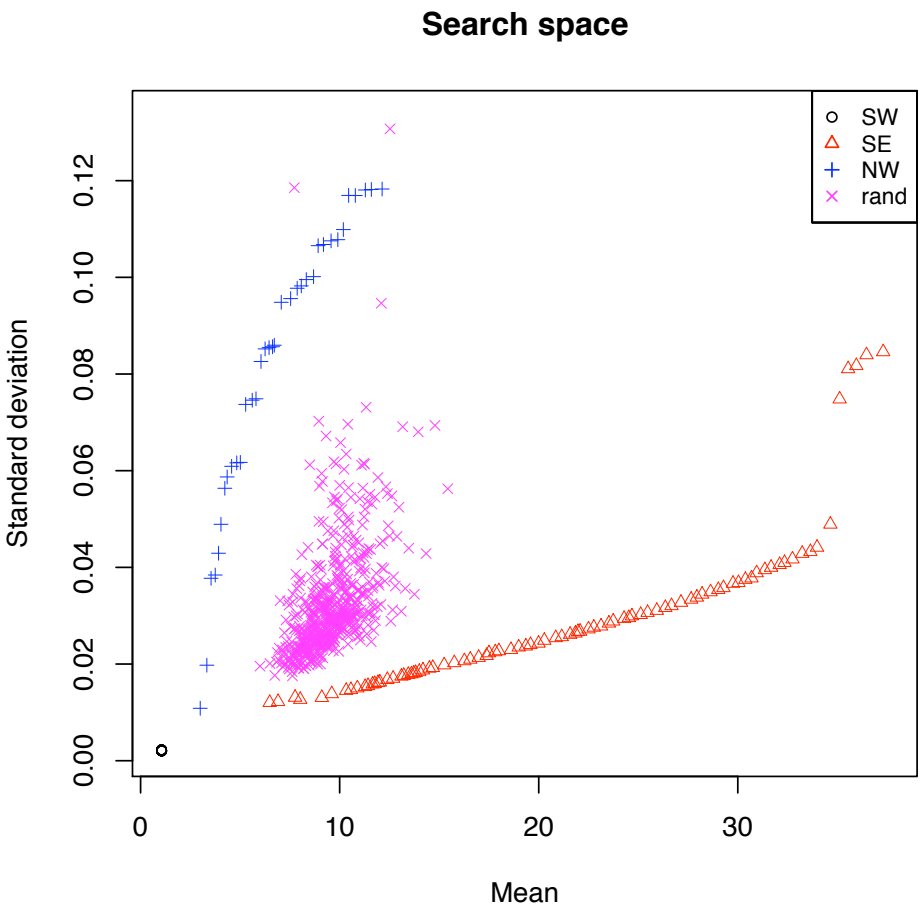


Figure 15: Mean vs. std. dev. of the makespan of different schedules in random and border cases, with a layrpred graph and TaskNumber = 1000

takes high values, *i.e.* when the uncertainty range of the durations is subject to large variations.

9.3 Average quality

The purpose of this section is to assess the quality of the schedules generated by the strategies presented in Section 7 and 8 in term of robustness and average performance. Figure 16 is a representative example regarding the position of the approximation sets (or Pareto fronts) of each strategy. Error bars denote the confidence intervals of each schedule evaluation.

Assessing the quality of an approximation set has been shown to be extremely difficult as several criteria can be measured for a given set (spread of the front, optimality, ...). Among the numerous existing quality indicators, we have chosen the binary ϵ-indicator which is presented and recommended by Zitzler *et al.* [46]. We compute it for each pair of heuristics (in Table 4). The value $I_\epsilon(A, B)$ corresponds to the ratio between a chosen solution of A and one of B for a selected criterion. It is roughly related to the relative distance be-

tween the two Pareto fronts and if we have $I_\epsilon(A,B) \leq 1$ and $I_\epsilon(B,A) > 1$, then the approximation set A is better than B. In other cases, the two fronts are incomparable.

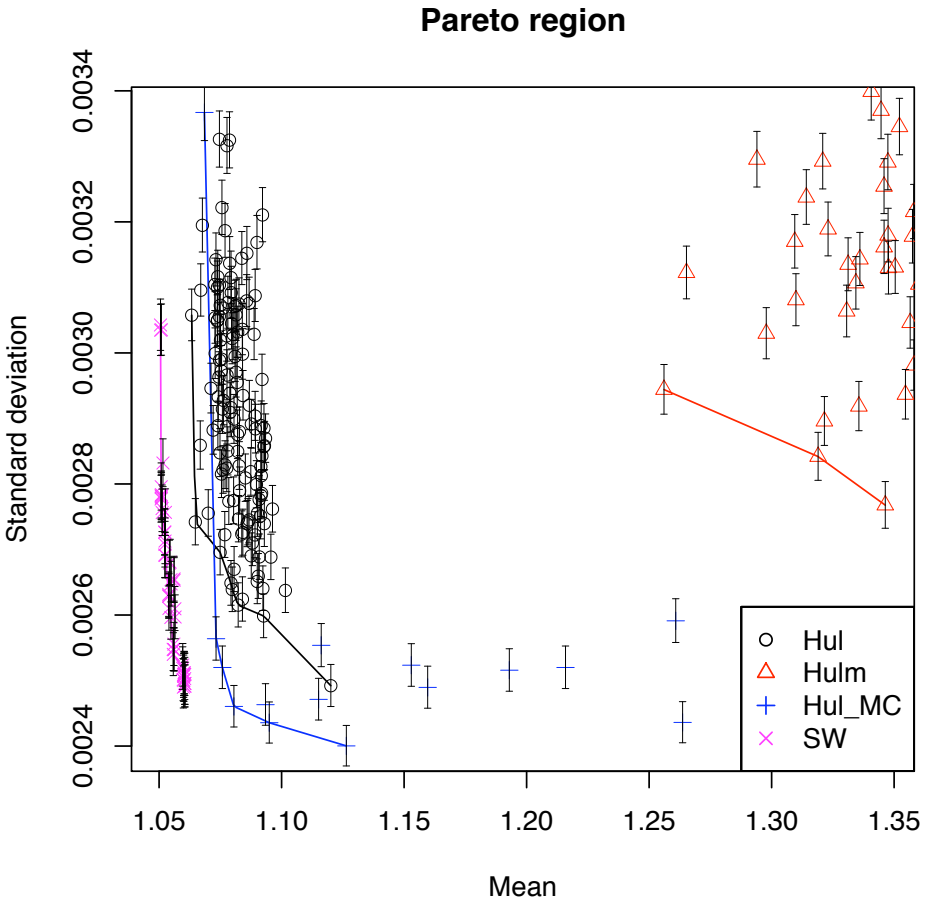


Figure 16: Mean vs. std. dev. of the makespan of different schedules in the Pareto region, with a layrpred graph and V_Cost = 1

B\A	Hul	Hulm	Hul_MC	SW
Hul	1.00	1.18	1.01	1.00
Hulm	0.90	1.00	0.87	0.90
Hul_MC	1.04	1.20	1.00	1.04
SW	1.04	1.20	1.02	1.00

Table 4: Comparison of the Pareto front: the binary ϵ-indicator values $I_\epsilon(A,B)$ for all pairs of strategies

It is clear in this example that *Hulm* performs the worst and *SW* is better than *Hul*. $I_\epsilon(\text{SW}, \text{Hul_MC}) > I_\epsilon(\text{Hul_MC}, \text{SW})$ does not necessary mean that *Hul_MC* is better than the MOEA. It can be explained by the fact that the indicator is a ratio between either 2 means or between 2 standard deviations.

Since this last criterion varies relatively the most, the presence of more robust solutions has more impact on the ϵ-indicator.

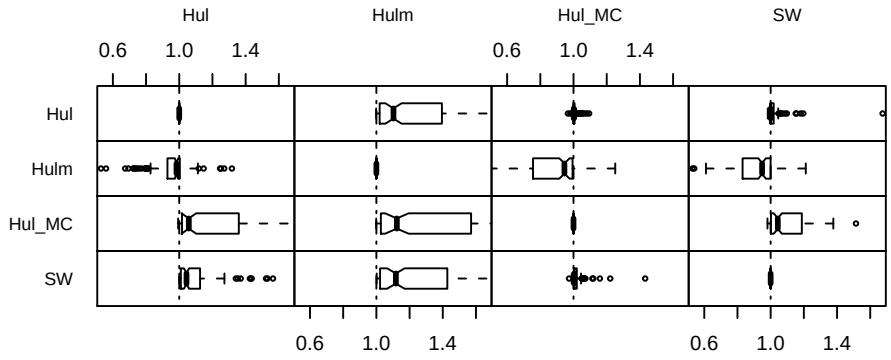


Figure 17: Comparison of the Pareto fronts: boxplot of the indicators over every experiment

Since the ϵ-indicator is a ratio, it allows doing comparison on every experiment scenario. Figure 17 represents the summary of every computed indicator in the form of boxplots. Boxplots allow to represent a five number summary of a given set (in this case, this is the 5th percentile, the first quartile, the median, the last quartile and the 95th percentile) and the outliers that exceed these values. For example, on the line *Hulm* and column *SW*, these 5 values are respectively 0.61, 0.83, 0.94, 1 and 1.21. We show that the previous remarks are general: *Hulm* is the worst heuristics (despite being the most sophisticated) and that the MOEA, *Hul* and *Hul_MC* are mostly incomparable. Although it does not appear on this figure, the MOEA systematically outperforms other heuristics in term of average makespan.

9.4 Computation time

We compute the average runtime of every heuristic over 5 runs with random graphs (Strassen graphs having different number of tasks). These measures are represented on Table 5. The second time includes the MC evaluation of the schedules (except for *Hul_MC* because it is already included and *SW* because it is negligible).

Task number	10	100	1000
Hul	0.4”/1”	19”/1.2’	5.7’/37.6’
Hulm	0.5”/1”	2’/4.6’	1h33’/2h05’
Hul_MC	1.1’	19.6’	3h24’
SW	55.8’	1h30’	6h44’

Table 5: Execution time of every strategy

It would have been possible to reduce the number of MC simulations in order to have similar execution time for *Hul* and *Hul_MC*. However, with 600 simulations, the standard deviation precision is about 25% which is quite high. Hence, *Hul_MC* is relevant only with high number of simulations.

9.5 MOEA evolution

Figure 18 illustrates the evolution of both criteria in function of the time taken by the MOEA when MC evaluations are used. The evolution of the standard deviation is less stable than with the average, which could be caused by a greater difficulty to generate robust schedules than efficient ones.

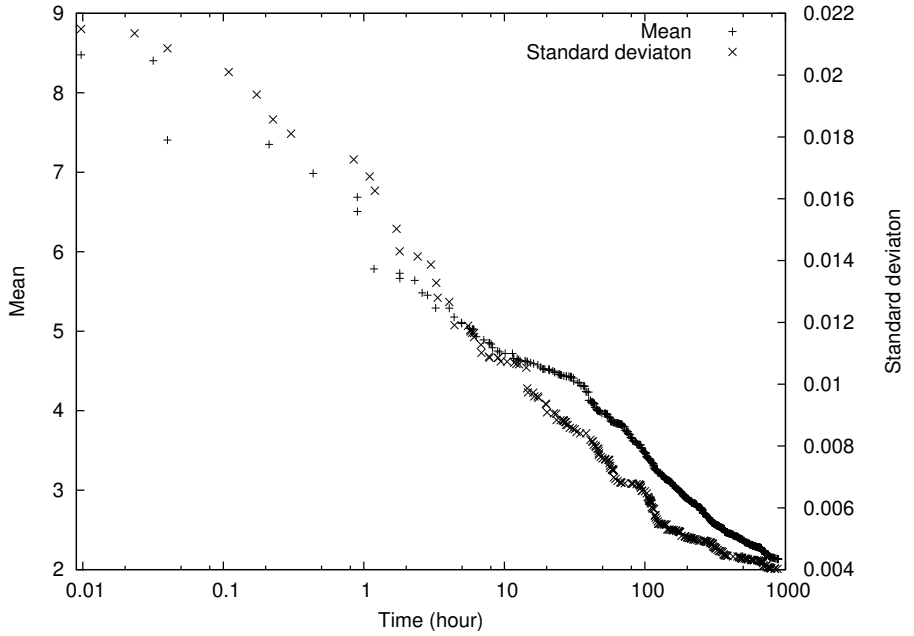


Figure 18: Evolution of the MOEA for both criteria with MC simulations

10 Conclusion

In this paper, we have studied the problem of scheduling a stochastic task graph with the goal of maximizing the robustness and minimizing the makespan on a heterogeneous environment.

This is a very difficult problem. Minimizing the makespan is an NP-hard problem while computing the metrics requires to solve a #P-complete problem which is very time-consuming.

Robustness is an objective that has led to a lot of different metrics. However, there is no consensus on a wide-accepted metric. In order to compare robustness metrics, we have proposed a comprehensive set of such metrics presented in the literature in the case of task graph scheduling with precedence constraints.

We have experimentally compared the robustness metrics on a set of different task graphs. This empirical study was intended to determine the relationship between these metrics. The first conclusion we can draw is that several of them are equivalent mostly due to the implication of the central limit theorem. Consequently, the simplest of these metrics, certainly the standard deviation, is sufficient in most real cases and denotes the absolute dispersion of the makespan,

its entropy, etc. (which are all related). An other important point is the unsuitability of the slack for our uncertainty model. We presented some arguments to justify this observation. Moreover, we have shown that in most cases the distribution of the makespan is sufficiently close to a Gaussian. This justifies our approximations used to evaluate this distribution in our heuristics.

Despite a fairly high correlation value between the makespan mean and standard deviation, most of the time there exists a set of Pareto-optimal solutions. This set tends to grow as the coefficient of variation of the uncertainty level (V_UL) is large. This justifies the design of bicriteria strategies.

Based on our understanding of the robustness metrics, we have proposed different strategies: three heuristics and a multi objective evolutionary algorithm (MOEA). The goal of having different strategies is to tackle the trade-off between computation time of the solutions and the quality of the solutions. Concerning our MOEA, we have proved its convergence by extending previous results on the global nature of the mutation operator.

We have then conducted an experimental study of our heuristics for the scheduling problem. The slowest strategy is our evolutionary algorithm. The proposed *Hul_MC* (an extension of the makespan-centric heuristic HEFT) is faster but provides worse makespan solution than our MOEA. *Hul* and *Hulm* are even faster heuristics but give even worse results for both criteria. All these strategies are also able to give a Pareto front of the solution. Therefore we are able to help the user in choosing another trade-off between makespan and robustness when necessary.

In future work we want to port the proposed solutions to real-world execution environments with concrete applications. In order to do that, we will need to analyze traces and to adapt our models and/or algorithms to the corresponding environments.

It is also important to remark that our method can be extended to other makespan-centric heuristics (BIL, PCT, HBMCT, CC, ILHA). Evaluation of these extensions is also left to future work. Moreover, we need a better evaluation mechanism for our MOEA in order to systematically outperform *Hul_MC* for the robustness. This requires either to better take into consideration the approximation of the current fitness function or to develop more precise evaluation methods.

The proposed scheduling heuristics are all static ones. This means that the scheduling decisions are made prior to the execution. Even, in the case where uncertainties can be very high we believe that it is important to have a good initial (static) solution even if on the fly dynamic adaptation can be made. However, coupling static solution with dynamic decision in order to adapt to the change is a difficult task and is left for future work.

11 Acknowledgement

We would like to thank Frédéric Suter for providing helpful comments on the content of this paper.

References

[1] Shoukat Ali, Anthony A. Maciejewski, Howard Jay Siegel, and Jong-Kook Kim. Measuring the Robustness of a resource Allocation. *IEEE Transaction on Parallel and Distributed Systems*, 15(7):630–641, July 2004.

[2] Shoukat Ali, Howard Jay Siegel, Muthucumaru Maheswaran, Debra Hensgen, and Sahra Ali. Representing Task and Machine Heterogeneities for Heterogeneous Computing Systems. *Tamkang Journal of Science and Engineering,* Special 50th Anniversary Issue, 3(3):195–207, November 2000.

[3] Ladislau Bölöni and Dan C. Marinesco. Robust scheduling of metaprograms. *Journal of Scheduling*, 5(5):395–412, September 2002.

[4] Sébastien Cahon, Nordine Melab, and El-Ghazali Talbi. ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. *Journal of Heuristics*, 10(3):357–380, 2004.

[5] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145–162, March/April 1961.

[6] David W. Corne, Nick R. Jerram, Joshua D. Knowles, and Martin J. Oates. PESA-II: Region-based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, San Francisco, California, USA, August 2001.

[7] Richard L. Daniels and Janice E. Carrillo. β-Robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29(11):977–985, November 1997.

[8] Andrew J. Davenport and J. Christopher Beck. A Survey of Techniques for Scheduling with Uncertainty. *Unpublished manuscript*, 2002.

[9] Andrew J. Davenport, Christophe Gefflot, and J. Christopher Beck. Slack-based Techniques for Robust Schedules. In *Proceedings of the Sixth European Conference on Planning (ECP-2001)*, pages 7–18, Toledo, Spain, September 2001.

[10] Kalyanmoy Deb, Amrit Pratab, Samir Agrawal, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[11] Bajis Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33(4):862–881, July 1985.

[12] Didier Dubois and Henri Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty.* Plenum Press, New York, 1988.

[13] Hesham El-Rewini, Theodore G. Lewis, and Hesham H. Ali. *Task Scheduling in Parallel and Distributed Systems.* Prentice Hall, 1994.

[14] Darin England, Jon Weissman, and Jayashree Sadagopan. A New Metric for Robustness with Application to Job Scheduling. In *14th IEEE International Symposium on High Performance Distributed Computing (HPDC-14)*, pages 135–143, July 2005.

[15] Helene Fargier, Philippe Fortemps, and Didier Dubois. Fuzzy scheduling: Modelling flexible constraints vs. coping with incomplete knowledge. *European Journal of Operational Research*, 147(2):231–252, 2003.

[16] Stefan Felsner and Klaus Reuter. The Linear Extension Diameter of a Poset. *SIAM Journal on Discrete Mathematics*, 12(3):360–373, 1999.

[17] Apostolos Gerasoulis, Jia Jiao, and Tao Yang. Experience with Graph Scheduling for Mapping Irregular Scientific Computation. In *First IPPS workshop on Solving Irregular Problems on Distributed Memory Machines*, pages 1–8, Santa Barbara, CA, USA, April 1995.

[18] GSL - GNU Scientific Library. http://www.gnu.org/software/gsl/.

[19] Jane N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18(2):139–147, 1998.

[20] Willy Herroelen and Roel Leus. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620, April 2004.

[21] Willy Herroelen and Roel Leus. Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165(2):289–306, September 2005.

[22] Yaochu Jin and Jürgen Branke. Evolutionary Optimization in Uncertain Environments–A Survey. *IEEE Transactions on Evolutionary Computation*, 9(3):303–3017, June 2005.

[23] Jr. John M. Burt and Mark B. Garman. Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis. *Management Science*, 18(3):207–217, November 1971.

[24] Seung-Jean Kim, Stephen P. Boyd, Sunghee Yun, Dinesh D. Patil, and Mark A. Horowitz. A Heuristic for Optimizing Stochastic Activity Networks with Applications to Statistical Digital Circuit Sizing. *Optimization and Engineering*, 8(4):397–430, December 2007.

[25] Erik Learned-Miller and Joseph DeStefano. A probabilstic upper bound on differential entropy. *IEEE Transactions on Information Theory*, Under revision, 2007.

[26] Joseph Y-T. Leung, editor. *Handbook of Scheduling*. Chapman & Hall/CCR, 2004.

[27] Yan Alexander Li and John K. Antonio. Estimating the Execution Time Distribution for a Task Graph in a Heterogeneous Computing System. In *6th IEEE Heterogeneous Computing Workshop (HCW'97)*, pages 172–184, Geneva, Switzerland, April 1997.

[28] J. W. S. Liu and C. L. Liu. Bounds on scheduling algorithms for heterogeneous computing systems. In *Proceedings of IFIP Congress 74*, pages 349–353, 1974.

[29] Arfst Ludwig, Rolf H. Mohring, and Frederik Stork. A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks. *Annals of Operations Research*, 102(1–4):49–64, February 2001.

[30] Muthucumaru Maheswaran and Howard Jay Siegel. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems. In *7th IEEE Heterogeneous Computing Workshop (HCW'98)*, pages 57–69, Orlando, Florida, USA, March 1998.

[31] Hyunok Oh and Soonhoi Ha. A Static Scheduling Heuristic for Heterogeneous Processors. In Luc Bougé, Pierre Fraigniaud, Anne Mignotte, and Yves Robert, editors, *Proceedings of the Second International Euro-Par Conference on Parallel Processing-Volume II*, volume 1124 of *Lecture Notes in Computer Science*, pages 573–577, Lyon, France, August 1996. Springer.

[32] Günter Rudolph. Convergence of Evolutionary Algorithms in General Search Spaces. In *International Conference on Evolutionary Computation*, pages 50–54, Nagoya, Japan, May 1996.

[33] Günter Rudolph and Alexandru Agapie. Convergence Properties of Some Multi-Objective Evolutionary Algorithms. In *Congress on Evolutionary Computation*, pages 1010–1016, La Jolla, California, USA, July 2000.

[34] Rizos Sakellariou and Henan Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In *13th IEEE Heterogeneous Computing Workshop (HCW'04)*, Santa-Fe, New Mexico, USA, April 2004.

[35] Johnatan Eliabeth Pecero Sanchez. *Local-Global scheduling interactions.* PhD thesis, Institut National Polytechnique de Grenoble, 2008.

[36] Vladimir Shestak, Jay Smith, Howard Jay Siegel, and Anthony A. Maciejewski. A Stochastic Approach to Measuring the Robustness of Resource Allocations in Distributed Systems. In *Proceedings of the 2006 International Conference on Parallel Processing (ICPP'06)*, pages 459–470, Columbus, Ohio, USA, August 2006.

[37] Zhiao Shi, Emmanuel Jeannot, and Jack J. Dongarra. Robust Task Scheduling in Non-Deterministic Heterogeneous Computing Systems. In *Proceedings of IEEE International Conference on Cluster Computing*, pages 1–10, Barcelona, Spain, September 2006. IEEE.

[38] Gilbert Sih and Edward Lee. A Compile-Time Scheduling Heuristic for Interconnection-Constrained Heterogenous Processor Architectures. *IEEE Transactions on Parallel and Distributed Systems*, 4(2):175–187, February 1993.

[39] Takao Tobita and Hironori Kasahara. A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *Journal of Scheduling*, 5(5):379–394, 2002.

[40] Haluk Topcuoglu, Salim Hariri, and Min-You Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. *Transactions on Parallel and Distributed Systems*, 13(3):260–274, March 2002.

[41] Richard M. van Slyke. Monte Carlo Methods and the PERT Problem. *Operations Research*, 11(5):839–860, September/October 1963.

[42] Lee Wang, Howard Jay Siegel, Vwani R. Roychowdhury, and Anthony A. Maciejewski. Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach. *Journal of Parallel and Distributed Computing*, 47(1):8–22, November 1997.

[43] Peter R. Wurman. Optimal Factory Scheduling Using Stochastic Dominance A*. In *13th National Conference on Artificial Intelligence*, page 1416, Portland, OR, USA, August 1996.

[44] E. Zitzler and S. Künzli. Indicator-Based Selection in Multiobjective Search. In *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, pages 832–842, Birmingham, UK, September 2004. Springer.

[45] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, September 2001.

[46] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, and Viviane Grunert da Fonseca. Performance Assessment of Multiobjective Optimizers: An Analysis and Review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, April 2003.



Centre de recherche INRIA Nancy – Grand Est
LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399